IN THE SPECIFICATION:

Please replace paragraph [0003] with the following:

[0003] To achieve fast performance and high availability to systems on a network, it is desirable to cache information close to the client applications that access the server information. Caching is used extensively on the Internet today, e.g. in proxy servers and client browsers, to reduce user-perceived latency and improve overall network performance. Co-pending, commonly-assigned utility application, "METHODANDAPPARATUS FOR OPTIMIZING QUERIES ON NETWORK DIRECTORY CACHES," SerialNo.09/448,811, filed on November24, 1999, (Received Notice of Abandonment dated 2/4/03) which is incorporated by reference herein, disclosed that cached directory entries can be reused in answering certain declarative queries. The use of semantic information in the client cache- although shown to be advantageous for efficient handling of declarative queries- also imposes a very high cost when individual user queries select just one or a few directory entries, as is often the case in many real LDAP applications. The storage overhead of maintaining the meta data that semantically describe the directory entries in the client cache becomes comparable to the size of the cached data, and the computational overhead of searching the meta data to determine if a user query can be answered from the cache becomes prohibitive.

Please replace paragraph [0017] with the following:

[0017] The passage staring at col. 2 line 37 FIG. 2 sets forth a flow chart illustrating the processing performed by the client in computing query templates, their costs, and maintaining their benefits, in accordance with a preferred embodiment of an aspect of the invention. Given a set of user LDAP queries, many possible query templates (each generalizing a different subset of the user queries) can be created. Keeping all possible templates can result in an inefficient use of the limited amount of cache space. Hence, at step 201, it is advantageous to generate a fixed number, say n, of query templates, referred to herein as "candidate" templates ("CT"). The candidate templates are kept as candidates to be admitted into the cache in the future. The number n of templates to be kept can be determined adaptively. Computation of a useful query template that generalizes a given pair of queries efficiently can take advantage of a combination of

techniques. For example, explicitly specified generalization hierarchies on attribute domains can be utilized, e.g., prefix matching on telephone numbers and suffix matching on email addresses, to compute generalizations of atomic filters. The atomic filters "(mail=olga@research.att.com)" and "(mail=divesh@research.att.com)" would generalize to the filter "(mail=*@research.att.com)". The natural hierarchy on conjunctive filters can also be utilized, based on the subset relationship between sets of conjuncts. For example, filters "(&(objectClass=lip) (mail=rng@research.att.com))" and "(& (mail=olga@research.att.com) (gender=f))" would generalize to the filter "(mail=*@research.att.com)". At step 202, a cost and benefit is associated with each such candidate template. Each candidate and actual query template t can be advantageously annotated with three statistical components: (i) s(t): size of the result of t, (ii) c(t): cost of execution of t, and (iii) b(t): benefit of caching t. The size s(t) can be efficiently estimated, without evaluating t at the directory server, based solely on the statistics maintained about the directory entries at the client. In particular, pruned suffix trees are very useful when estimating the sizes of string wildcard queries that constitute query templates. See co-pending, commonly-assigned ~~utility application~~ US Patent 6,401,088, titled "METHOD AND APPARATUS FOR SUBSTRING SELECTIVITY ESTIMATION," ~~Serial No. 09/476,71 5, filed on December 30, 1999,~~ which is incorporated by reference herein. The cost c(t) is a measure of the total evaluation cost of the query template at the directory server, and the communication cost of transmitting the query and the query answer over the network. This can be estimated at the client using knowledge of the network and directory server parameters. One would ideally like to measure the benefit of a template *t* as the sum of the costs $c(q_i)$ of future user queries q_i that could be answered using the result of *t*. Since future reference patterns are not available in advance, the probability of a future reference can be approximated from a past reference pattern using the history of user queries. However, maintaining the entire history of user queries is infeasible. Hence, an estimate of the benefit of a template *t* can be computed using the benefits of available candidate templates that instantiate *t*. For a template *t* that is not instantiated by any candidate templates, the benefit b(*t*) can be estimated by its cost c(*t*). These three components constitute a "profit" metric that the replacement policies can use to find the most profitable templates to cache. As

A2 demonstrated later, a useful profit p(t) of a template t is computed as: p(t) = ((b(t)-c(t))/s(t).